Solving Hard Multiobjective Optimization Problems using ε-Constraint with Cultured Differential Evolution

Ricardo Landa Becerra and Carlos A. Coello Coello

Evolutionary Computation Group at CINVESTAV-IPN (EVOCINV)
Electrical Eng. Department, Computer Science Dept.
Av. IPN No. 2508 Col. San Pedro Zacatenco, México D.F. 07300, MÉXICO
rlanda@computacion.cs.cinvestav.mx
ccoello@cs.cinvestav.mx

Abstract. In this paper, we propose the use of a mathematical programming technique called the ε-constraint method, hybridized with an evolutionary single-objective optimizer: the cultured differential evolution. The ε-constraint method uses the cultured differential evolution to produce one point of the Pareto front of a multiobjective optimization problem at each iteration. This approach is able to solve difficult multiobjective problems, relying on the efficiency of the single-objective optimizer, and on the fact that none of the two approaches (the mathematical programming technique or the evolutionary algorithm) are required to generate the entire Pareto front at once. The proposed approach is validated using several difficult multiobjective test problems, and our results are compared with respect to a multi-objective evolutionary algorithm representative of the state-of-the-art in the area: the NSGA-II.

1 Introduction

Evolutionary multiobjective optimization consists of using evolutionary algorithms to solve problems with two or more (often conflicting) objective functions. This research area has become very popular in the last few years [1]. Concurrently, more challenging problems have been integrated into the most recent benchmarks, some of which require a considerably high number of objective function evaluations to be solved, or can even make current algorithms to fail in their efforts to generate the true Pareto front [2].

The ε-constraint method is a mathematical programming technique, which transforms a multiobjective optimization problem into several constrained single-objective problems. This method has not been used too often in evolutionary computation, due to the fact that it does not generates a set of nondominated solutions in a single run, as most evolutionary algorithms do. Moreover, it has been found that this method is relatively expensive when solving "easy" multiobjective problems, because of the several single-objective optimizations executed.

In this paper we propose the use of the ε-constraint method together with an efficient evolutionary approach which solves constrained single-objective optimization problems. The optimizer consists of a differential evolution-based cultural algorithm, which improves the optimization process by means of domain information extracted during the evolutionary search.

The rest of the paper is organized as follows: in Section 2, the ε-constraint method is presented in some detail. Section 3 briefly describes the cultured differential evolution approach, which is the single-objective optimizer adopted in this work. Section 4 describes our proposed approach. Section 5 contains our results, and a comparative study with respect to the NSGA-II. Finally, Section 6 provides our conclusions and some possible paths for future research.

2 The ε-Constraint Method

This is a multiobjective optimization technique, proposed by Haimes et al. [3], for generating Pareto optimal solutions. It makes use of a single-objective optimizer which handles constraints, to generate one point of the Pareto front at a time. For transforming the multiobjective problem into several single-objective problems with constraints it uses the following procedure (assuming minimization for all the objective functions):

$$\begin{aligned} &\text{minimize} \quad f_l(\mathbf{x}) \\ &\text{subject to } f_j(\mathbf{x}) \leq \varepsilon_j \text{ for all } j = 1, 2, \ldots, m,\ j \neq l, \\ &\qquad\qquad \mathbf{x} \in S \end{aligned}$$

where $l \in \{1, 2, \ldots, m\}$ and S is the feasible region, which can be defined by any equality and/or inequality constraint. The vector of upper bounds, $\varepsilon = (\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_m)$, defines the maximum value that each objective can have. In order to obtain a subset of the Pareto optimal set (or even the entire set, in case this set is finite), one must vary the vector of upper bounds along the Pareto front for each objective, and make a new optimization process for each new vector. The generation of different points of the Pareto front using different values of the upper bound is illustrated in Figure 1.

For any nonlinear multiobjective optimization problem, the solution of an ε-constraint problem yields a weakly Pareto optimal solution [3]. A true Pareto optimal solution can be obtained either if the solution is unique, or if the optimizations are done for all the objectives before reporting the solution [4]. However, to improve the speed of the generation of solutions, only one optimization per point can be performed to obtain an approximation of the Pareto optimal set.

To the best of our knowledge, the only attempt to hybridize the ε-constraint method with an evolutionary algorithm is the approach called CMEA [5]. This approach performs the intermediate optimizations using a standard evolutionary algorithm. To reduce the computational cost of each independent optimization, the final population of one optimization process is used as the initial population for the next one; however, the authors noted the lack of diversity of the

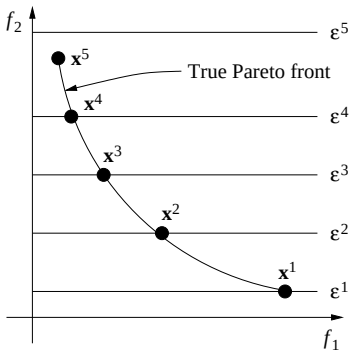

Fig. 1. Generating different solutions with the ε-constraint method

approach and proposed a high mutation rate at the beginning of each process. The authors provide no further details about the mechanism adopted to handle the constraints in the single-objective optimizer.

In [6], the authors proposed an extension of CMEA for three-objective problems. However, the algorithm does not seem able to find the extreme points of the Pareto front itself, since they are provided *a priori* by the user. Regarding the number of fitness function evaluations needed for CMEA to obtain good results, in [6], the authors mention that they perform 500,000 evaluations for the three-objective knapsack problem.

3 Cultured Differential Evolution

The cultured differential evolution is a cultural algorithm [7] based on differential evolution [8], designed to solve nonlinear constrained optimization problems. In previous experiments [9], this algorithm exhibited a very good performance, obtaining competitive results when compared to other state-of-the-art evolutionary optimization techniques, but requiring only a fraction of their fitness function evaluations. This is because of the use of domain knowledge, extracted during the evolutionary process, to efficiently guide the search. Next, we will briefly describe this approach.

Cultural algorithms are made of two main components: The *population space* consists of a set of possible solutions to the problem, and can be modeled using any population based technique. The *belief space* is the information repository in which the individuals can store their experiences for the other individuals to learn them indirectly; it may be composed by several *knowledge sources*. Our proposed approach uses differential evolution in the population space [8]. A pseudo-code of our approach is shown in Algorithm 1.

In the initial steps of the algorithm, a population of $popsize$ individuals, $\mathbf{x}^j, j = 1, \ldots, popsize$, is created; each individual contains the n parameters of the problem, $\mathbf{x}^j = (x_1^j, \ldots, x_n^j)$. An initial belief space is also created. For the

Algorithm 1 Pseudo-code of the cultured differential evolution

```
Generate initial population of size popsize
Initialize the belief space
repeat
  for each individual j in the population do
    Randomly select a knowledge source k_s from the belief space
    Generate a random integer i_rand ∈ (1, n)
    for each parameter i do
      x_i^{j'} = { influence(k_s) if rand(0,1) < CR or i = i_rand
                 { x_i^j          otherwise
    end for
    Replace x^j with the child x^{j'}, if x^{j'} is better
  end for
  Update the belief space
until the termination condition is achieved
```

offspring generation, the variation operator of differential evolution is modified by the $influence()$ function of a knowledge source, but the parameters CR and F of the standard differential evolution are also required. To determine if a child is better than its parent, and, therefore, if it can replace it, we use the following rules: 1. A feasible individual is better than an infeasible one. 2. If both are feasible, the individual with the best objective function value is better. 3. Otherwise, the individual with less amount of constraint violation is considered better. The amount of constraint violation is measured using the expression: $viol(\mathbf{x}) = \sum_{c=1}^{C} \frac{g_c(\mathbf{x})}{g_{maxc}}$ where $g_c(\mathbf{x})$ with $c = 1, \ldots, C$ are the constraints of the problem, and g_{maxc} is the largest violation found for the constraint $g_c(\mathbf{x})$ so far.

In our approach, the belief space is divided into 4 knowledge sources:

Situational Knowledge: consists of the best exemplar found along the evolutionary process. Its infuence function modifies the direction of the variation operator to follow the leader.

Normative Knowledge: contains the intervals for the decision variables where good solutions have been found, to move new solutions towards them, through the use of its influence function.

Topographical Knowledge: It consists of a set of cells, and the best individual found on each cell. The topographical knowledge has an ordered list of the best cells, based on the fitness value of the best individual on each of them. Its influence function moves newly generated individuals towards the best cells.

History Knowledge: was originally proposed for dynamic objective functions [10]. It records in a list, the location of the best individual found before each environmental change. In our approach, instead of detecting changes of the environment, we use it to escape from local optima.

At the beginning, all the knowledge sources have the same probability to be applied, but during the evolutionary process, the probability of applying each knowledge source is updated according to its success rate.

4 Hybridizing the ε-Constraint Method with CDE

There are two main possibilities of how we can vary the ε values: one is to have an approximation of the dimensions of the Pareto front, and then divide it into a number of intervals depending of the number of solutions that we want as outcome. The other, proposed by Laumanns et al. [11] is to execute an initial optimization without constraints, and then use the result of this first step to set the values for ε. If the Pareto front is discrete, this approach is particularly suitable, because it can find the entire Pareto optimal set, as proved in [11].

As our proposed approach is designed to deal with real-valued problems, it is most likely to have a continuous Pareto front, so we chose the first approach (from the two previously mentioned) to obtain ε. The ε_j must vary from the best to the worst value for the objective j, *i.e.* the search must move from the ideal to the nadir objective vector. The estimation of the ideal objective vector involves individual optimizations of one objective at a time. On the other hand, the estimation of the nadir objective vector is a more difficult task [4]. Currently, there are no efficient and reliable methods to estimate the nadir point, for an arbitrary problem. Only for the two-objective case, there exists a simple method that can provide a good estimation, which is called the *payoff table*. Due to this limitation, the proposed approach is currently working only for two-objective problems. However, there are very hard two-objective problems in the literature, which are very difficult to solve efficiently by any of the current multi-objective evolutionary algorithms (MOEAs). Some details about the estimation of the dimensions of the Pareto front, in the proposed approach, are shown in the first steps of Algorithm 2.

The single-objective optimizer, in which our method is based, is the cultured differential evolution previously described. Let's now assume that it is available as the procedure $cde(f_l, \varepsilon, g)$, which performs the optimization process of the ε-constraint method during g generations and returns the best point found. If the procedure is called without any ε values ($cde(f_l, g)$), the optimization is performed removing the constraints of the form $f_j(\mathbf{x}) \leq \varepsilon_j$. The pseudo-code of the ε-constraint with CDE (ε-CCDE) is shown in Algorithm 2.

In Algorithm 2, the lower and upper bounds, lb and ub, are increased by a tolerance t; this is done since the results of the *cde* procedure are only approximations, and it is possible to find a better point outside of them. We use $t = 0.05(ub - lb)$. The ε values are updated with a δ, which is dependent of the number of points in the Pareto front desired by the user or the decision maker, p. It is obtained as follows: $\delta = \frac{ub-lb}{p}$ This way, we aim that the final points are equally spaced in their projection over the f_2 axis. g is an input parameter of the algorithm, but it is very important, because together with p and the population size of the *cde* procedure, $popsize$, define the total number of fitness function evaluations required for the approach. The number of fitness function evaluations is approximately $p \cdot g \cdot popsize$.

Algorithm 2 shows f_1 as the objective to be optimized, and f_2 as the constraint. However, one can interchange the roles of the objectives if the problem looks harder to solve in the original setting. In the experiments shown in this

Algorithm 2 ε-Constraint with CDE.

$P = \emptyset$
$ub = f_2(cde(f_1, 2g))$
$lb = f_2(cde(f_2, 2g))$
$ub = ub + t$, $lb = lb - t$
$\varepsilon = lb$
while $\varepsilon \leq ub$ **do**
  $\mathbf{x} = cde(f_1, \varepsilon, g)$
  if $\mathbf{x}$ is nondominated with respect to P **then**
    $P = P - \{\mathbf{y} \in P \mid \mathbf{x} \succ \mathbf{y}\}$
    $P = P \cup \{\mathbf{x}\}$
  end if
  $\varepsilon = \varepsilon + \delta$
end while

paper, the original setting was always preserved, and f_1 was always taken as the objective to optimize, to allow a fair comparison. In order to improve the performance of each optimization process, the algorithm shares a percentage of the population, in the initial population of the next process. This helps because the problems to be solved are very similar, and the only change is the upper bound of the objective functions that are treated as constraints. When all the population is shared, the loss of diversity leads to premature convergence. In practice, we found that a small percentage (around 10%) of the population to be shared is enough to improve convergence without losing diversity.

5 Comparison of Results

In order to validate the performance of the proposed approach, some test functions have been taken from the specialized literature. One may think that the several single-objective optimizations required may give rise to a prohibitively high computational cost, which is unnecessary considering that a modern MOEA may produce a similar approximation of the Pareto front at a much more affordable computational cost. There are problems, however, where this is not the case, and in which a modern MOEA cannot converge to the true Pareto front even if we do not restrict the number of evaluations performed. It is precisely in those cases for which we believe that our approach can be a viable alternative.

In order to validate our hypothesis we looked specifically for hard multiobjective problems within the existing benchmarks. Our search led us to the use of DTLZ8 and DTLZ9 (from [12]), with 20 decision variables, as suggested by their designers. The main difficulty of these two problems lies on the satisfaction of their constraints. We also looked at a more recent benchmark proposed by Huband et al. [2], where we found harder problems (WFG1, WFG2, WFG3 and WFG9). Each of them has 24 variables. WFG1 is strongly biased toward small values of the first 4 variables, WFG2 and WFG3 are non-separable, but WFG2 has also a disconnected Pareto front, and WFG9 is a deceptive problem.

We decided to compare results with respect to the NSGA-II [13], since this is an approach representative of the state-of-the-art in the area.

5.1 Experimental Setup

We ran both algorithms during 50,000 fitness function evaluations each (except for two problems). We aimed to obtain a set of 50 points as a result of each run, so we adapted the parameters according to that. For the ε-CCDE, the parameters adopted were: $p = 50$, $g = 48$, with 10% of the population shared between optimizations (this 10% is chosen at random). For the *cde* procedure we used $popsize = 20$, $F = 0.7$, $CR = 0.5$. The population size of NSGA-II was set to 52, and the number of generations to 962. The rest of the parameteres were set as recommended by its authors: probability of crossover = 0.9, probability of mutation = 0.0333, the value of the distribution index for crossover = 15, and the value of the distribution index for mutation = 20.

Only for WFG1, the total number of fitness function evaluations was increased to 250,000, because this is a really difficult problem. The parameters adopted in this case were: $g = 120$ and $popsize = 40$. The number of generations of the NSGA-II was changed in this case to 4808. Even with this large number of iterations, the NSGA-II was not able to reach the true Pareto front. On the other hand, for WFG2, the total number of fitness function evaluations was decreased to 25,000, because this problem is less difficult than the others. The NSGA-II ran in this case for 481 generations, and our approach adopted $popsize = 10$.

In Figure 2, we show the results of a single run for each test problem. Since a visual comparison of the results may be inaccurate, we also used some performance measures to allow a quantitative comparison of results.

5.2 Performance Measures

To assess the performance of the proposed approach, we adopted the two set coverage (CS) metric [14], which is an indicator of how much a set covers (or dominates) another one. A value of $CS(X,Y) = 1$ means that all points in X dominate or are equal to Y. If $CS(X,Y) = 0$, there are no points in X that dominate some point in Y. We executed our ε-CCDE 30 times per problem, and then executed the NSGA-II 30 times with the same random seeds, and we performed 30 one-to-one comparisons. The results are summarized in Table 1.

In all the problems in Table 1, the ε-CCDE obtained better average values. However, the improvement is not always the same. In WFG1, all the points of ε-CCDE always dominate the points produced by the NSGA-II, because the latter cannot properly converge. On the other hand, in DTLZ8 and 9, both algorithms could make most of their points to converge to the true Pareto front. But, as it can be seen from the application of the next performance measure, the ε-CCDE was able to generate points nearer to the ends of the true Pareto front.

Our second performance measure was the binary coverage (Q_c) [15], which is an indicator of the ability of an algorithm to obtain solutions near the extrema of the Pareto front, measuring the largest possible angle between two vectors of the

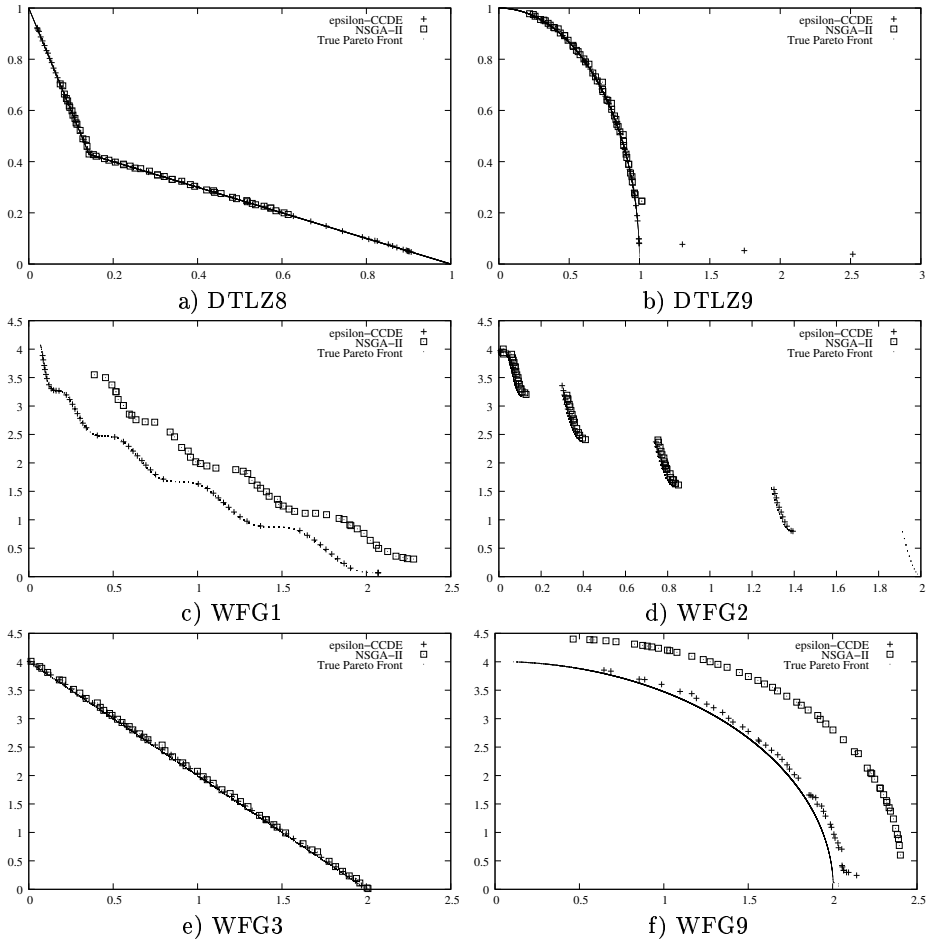

Fig. 2. Results produced by our ε-CCDE and the NSGA-II on the six test problems adopted.

Table 1. Mean and standard deviation of the CS measure (a larger value is better for the first algorithm)

Test Problem	CS(ε-CCDE, NSGA-II) mean (std. dev.)	CS(NSGA-II, ε-CCDE) mean (std. dev.)
DTLZ8	0.1415 (0.0521)	0.0185 (0.0184)
DTLZ9	0.1849 (0.0715)	0.1334 (0.0805)
WFG1	1.0000 (0.0000)	0.0000 (0.0000)
WFG2	0.8509 (0.1771)	0.0362 (0.0614)
WFG3	0.3987 (0.2691)	0.0908 (0.1368)
WFG9	0.6415 (0.3669)	0.0995 (0.2114)

output of an algorithm. This is a second criterion when proper convergence has been achieved. A value of $Q_c(X,Y) > 0$ means that X obtained points nearer to the extrema of the Pareto front (it covers a larger angle). In Table 2, we show the results (note that $Q_c(Y,X) = -Q_c(X,Y)$).

Table 2. Mean and standard deviation of the binary coverage measure (a larger value is better for the first algorithm)

Test Problem	Q_c(ε-CCDE, NSGA-II) mean (std. dev.)
DTLZ8	0.2496 (0.0536)
DTLZ9	0.1204 (0.1180)
WFG1	0.2112 (0.0634)
WFG2	0.0677 (0.2172)
WFG3	-0.0299 (0.0253)
WFG9	-0.0913 (0.1154)

This time, our approach obtained the largest values for DTLZ8, DTLZ9 and WFG1. For WFG3 and WFG9, this metric indicates that the NSGA-II can cover a larger portion of the Pareto front. However, it is important to keep in mind that this is a secondary criterion, which becomes relevant only when convergence has been achieved. In this case, and based on the two set coverage measure, our ε-CCDE achieved a better convergence than the NSGA-II.

6 Conclusions and Future Work

In this paper, we explored the use of the ε-constraint method hybridized with an efficient evolutionary single-objective optimizer, when solving hard multiobjective optimization problems. Our results show that the proposed approach can solve problems that a highly competitive MOEA (the NSGA-II) cannot. Also, there are some problems where the NSGA-II can converge properly, but it cannot reach the ends of the true Pareto front, while our proposed approach obtained a better spread of solutions in such cases. This approach may be recommended when other algorithms cannot achieve a proper convergence, or when it is known that the problem is deceptive or is strongly biased.

As part of our future work, we aim to extend our approach for $m > 2$ objectives. This task requires that we implement a mechanism to estimate the ideal and nadir objective vectors for more than 2 objectives.

Acknowledgments

The first author acknowledges support from CONACyT through a scholarship to pursue graduate studies at CINVESTAV-IPN. The second author acknowledges support from CONACyT through project number 45683-Y.

References

1. Coello Coello, C.A., Van Veldhuizen, D.A., Lamont, G.B.: Evolutionary Algorithms for Solving Multi-Objective Problems. Kluwer Academic Publishers, New York (2002) ISBN 0-3064-6762-3.
2. Huband, S., Barone, L., While, L., Hingston, P.: A Scalable Multi-objective Test Problem Toolkit. In Coello Coello, C.A., et al., eds.: Evolutionary Multi-Criterion Optimization. Third International Conference, EMO 2005, Guanajuato, México, Springer. LNCS Vol. 3410 (2005) 280–295
3. Haimes, Y.Y., Lasdon, L.S., Wismer, D.A.: On a Bicriterion Formulation of the Problems of Integrated System Identification and System Optimization. IEEE Transactions on Systems, Man, and Cybernetics **1**(3) (1971) 296–297
4. Miettinen, K.M.: Nonlinear Multiobjective Optimization. Kluwer Academic Publishers, Boston, Massachusetts (1999)
5. Ranjithan, S.R., Chetan, S.K., Dakshima, H.K.: Constraint Method-Based Evolutionary Algorithm (CMEA) for Multiobjective Optimization. In Zitzler, E., et al., eds.: First International Conference on Evolutionary Multi-Criterion Optimization. Springer-Verlag. LNCS No. 1993 (2001) 299–313
6. Kumar, S.V., Ranjithan, S.R.: Evaluation of the Constraint Method-Based Evolutionary Algorithm (CMEA) for a Tree-Objective Optimization Problem. In Langdon, W., et al., eds.: Genetic and Evolutionary Computation Conference (GECCO'2002), San Francisco, California, Morgan Kaufmann Publishers (2002) 431–438
7. Reynolds, R.G.: An Introduction to Cultural Algorithms. In Sebald, A.V., Fogel, L.J., eds.: Third Annual Conference on Evolutionary Programming. World Scientific, River Edge, New Jersey (1994) 131–139
8. Price, K.V.: An introduction to differential evolution. In Corne, D., et al., eds.: New Ideas in Optimization. McGraw-Hill, London, UK (1999) 79–108
9. Landa Becerra, R., Coello Coello, C.A.: Optimization with Constraints using a Cultured Differential Evolution Approach. In Beyer, H.G., et al., eds.: Genetic and Evolutionary Computation Conference (GECCO'2005). Volume 1., Washington, D.C., U.S.A., ACM Press (2005) 27–34
10. Saleem, S.M.: Knowledge-Based Solution to Dynamic Optimization Problems using Cultural Algorithms. PhD thesis, Wayne State University, Detroit, Michigan (2001)
11. Laumanns, M., Thiele, L., Zitzler, E.: An efficient, adaptive parameter variation scheme for metaheuristics based on the epsilon-constraint method. European Journal of Operational Research **169** (2006) 932–942
12. Deb, K., Thiele, L., Laumanns, M., Zitzler, E.: Scalable Test Problems for Evolutionary Multiobjective Optimization. In Abraham, A., et al., eds.: Evolutionary Multiobjective Optimization. Theoretical Advances and Applications. Springer, USA (2005) 105–145
13. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A Fast and Elitist Multiobjective Genetic Algorithm: NSGA–II. IEEE Transactions on Evolutionary Computation **6**(2) (2002) 182–197
14. Zitzler, E., Deb, K., Thiele, L.: Comparison of Multiobjective Evolutionary Algorithms: Empirical Results. Evolutionary Computation **8**(2) (2000) 173–195
15. Farhang-Mehr, A., Azarm, S.: Minimal Sets of Quality Metrics. In Fonseca, C.M., et al., eds.: Evolutionary Multi-Criterion Optimization. Second International Conference, EMO 2003, Faro, Portugal, Springer. LNCS. Volume 2632 (2003) 405–417